IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application No.: 10/602,551	§	Examiner: Dao, Thuy Chan
Filed: June 24, 2003	§	Group/Art Unit: 2192
Inventor(s):	§	Atty. Dkt. No: 5150-80201
Thomas A. Makowski, Rajesh Vaidya, Deborah E. Bryant and Brian M. Johnson	§	
Title: TASK BASED POLYMORPHIC GRAPHICAL PROGRAM FUNCTION NODES	§	

APPEAL BRIEF

Box: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir/Madam:

Further to the Notice of Appeal filed June 10, 2008, Appellant presents this Appeal Brief. Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by National Instruments Corporation, a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 11500 N. MoPac Expressway, Bldg. B, Austin, Texas 78759-3504.

II. RELATED APPEALS AND INTERFERENCES

No related appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-68 have been cancelled. Claims 69-92 are pending in the case. Claims 69-92 stand rejected under 35 U.S.C. § 102(b) and are the subject of this appeal. A copy of claims 69-92, incorporating entered amendments, as on appeal, is included in the Claims Appendix hereto.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been filed subsequent to the rejection in the Final Office Action of March 10, 2008. The Claims Appendix hereto reflects the current state of the claims.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The present application relates to the field of graphical programming, and more particularly to polymorphic graphical program function nodes in a graphical programming language. A palette for displaying these nodes is also presented.

Independent claim 69 recites a computer-accessible memory medium that stores program instructions executable by a processor to configure a graphical program function node with graphical program code implementing functionality in accordance with a user-selected function. *See, e.g., p.33:7-3; Figure 6.* A node is displayed in a graphical program. For example, the node may be displayed in the graphical program in response to user input, e.g., via dragging and dropping the node from a palette into the graphical program. *See, e.g., p.36:22-28, p.38:11-25.* First user input invoking display of a plurality of functions for the node is received. *See, e.g., p.38:28-p.39:3.* The plurality of functions for the node is displayed in response to the first user input. *See, e.g., p.39:5-23; Figure 7A, Figure 8.* Second user input selecting a function from the plurality of functions is received. *See, e.g., p.40:5-19, Figure 7A, Figure 8.* Graphical program code is determined based on the second user input, where the determined graphical program code includes a graphical representation of an implementation of the selected function, and where the determined graphical program code is executable to provide functionality in accordance with the selected function. *See, e.g., p.40:29-p.41:8.* The determined graphical program code is then associated with the node, where, when the node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function. For example, prior to the associating, the node may be a generic node, e.g., a generic read or write node (or other type of generic function node), while after the associating, the node may be a specific read or write node (or other type of specific function node). *See, e.g., p.41:10-p.43:5.*

Independent claim 77 recites a computer-implemented method for configuring a graphical program node, similar to the functionality recited in claim 69, summarized

above. *See, e.g., p.33:7-3; Figure 6.* A node is displayed in a graphical program. For example, the node may be displayed in the graphical program in response to user input, e.g., via dragging and dropping the node from a palette into the graphical program. *See, e.g., p.36:22-28, p.38:11-25.* First user input invoking display of a plurality of functions for the node is received. *See, e.g., p.38:28-p.39:3.* The plurality of functions for the node is displayed in response to the first user input. *See, e.g., p.39:5-23; Figure 7A, Figure 8.* Second user input selecting a function from the plurality of functions is received. *See, e.g., p.40:5-19; Figure 7A, Figure 8.* Graphical program code is determined based on the second user input, where the determined graphical program code includes a graphical representation of an implementation of the selected function, and where the determined graphical program code is executable to provide functionality in accordance with the selected function. *See, e.g., p.40:29-p.41:8.* The determined graphical program code is then associated with the node, where, when the node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function. For example, prior to the associating, the node may be a generic node, e.g., a generic read or write node (or other type of generic function node), while after the associating, the node may be a specific read or write node (or other type of specific function node). *See, e.g., p.41:10-p.43:5.*

Independent claim 85 is directed to a computer-accessible memory medium that stores program instructions executable by a processor to replace a first graphical program function node with a second graphical program function node in accordance with a user-selected function for the node. *See, e.g., p.43:7-p.44:2; Figure 6.* A node is displayed in a graphical program. For example, the node may be displayed in the graphical program in response to user input, e.g., via dragging and dropping the node from a palette into the graphical program. *See, e.g., p.36:22-28, p.38:11-25.* First user input invoking display of a plurality of functions for the node is received. *See, e.g., p.38:28-p.39:3.* The plurality of functions for the node is displayed in response to the first user input. *See, e.g., p.39:5-23; Figure 7A, Figure 8.* Second user input selecting a function from the plurality of functions is received. *See, e.g., p.40:5-19; Figure 7A, Figure 8.* A second node is determined based on the selected function, wherein the second node is or includes

a graphical representation of an implementation of the selected function, and wherein the second node comprises graphical program code executable to provide functionality in accordance with the selected function. *See, e.g., p.43:7-12.* The node in the graphical program is then replaced with the second node, where when the second node in the graphical program executes the graphical program code of the second node executes to provide the functionality in accordance with the selected function. *See, e.g., p.43:7-p.44:7.*

Independent claim 89 recites a computer-implemented method for configuring a graphical program node, similar to the functionality recited in claim 85, summarized above. *See, e.g., p.43:7-p.44:2; Figure 6.* A node is displayed in a graphical program. For example, the node may be displayed in the graphical program in response to user input, e.g., via dragging and dropping the node from a palette into the graphical program. *See, e.g., p.36:22-28, p.38:11-25.* First user input invoking display of a plurality of functions for the node is received. *See, e.g., p.38:28-p.39:3.* The plurality of functions for the node is displayed in response to the first user input. *See, e.g., p.39:5-23; Figure 7A, Figure 8.* Second user input selecting a function from the plurality of functions is received. *See, e.g., p.40:5-19; Figure 7A, Figure 8.* A second node is determined based on the selected function, where the second node is or includes a graphical representation of an implementation of the selected function, and where the second node includes graphical program code executable to provide functionality in accordance with the selected function. *See, e.g., p.43:7-12.* The node in the graphical program is then replaced with the second node, where when the second node in the graphical program executes the graphical program code of the second node executes to provide the functionality in accordance with the selected function. *See, e.g., p.43:7-p.44:7.*

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 69-92 were rejected under 35 U.S.C. 102(b) as being anticipated by Kudukoli (US Pat. Pub. No. 2001/0024211 A1).

VII. ARGUMENT

First Ground of Rejection

Claims 69-92 were rejected under 35 U.S.C. 102(b) as being anticipated by Kudukoli (US Pat. Pub. No. 2001/0024211 A1). Appellant respectfully traverses this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 69, 77

Independent claim 69 is separately patentable because the cited reference does not teach or suggest the limitations recited in this claim. For example, Appellant respectfully submits that Kudukoli fails to teach or suggest **associating the determined graphical program code with the node, wherein, when the node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function**, as recited in claim 69.

Cited Figure 4 and paragraphs [0100] – [0113] are directed to a method for programmatically creating a graphical program, specifically, via creation of a graphical program generation (GPG) program, typically by a user, after which the GPG program is executed. At runtime, the GPG program receives program information (via node inputs) specifying functionality for a new graphical program, and in response to this program information, the GPG program programmatically generates the new graphical program (or graphical program portion) implementing the specified functionality.

Cited Figure 23 and [0273] – [0275] disclose a graphical program that includes a user interface panel and a block diagram coupled to the user interface panel, where the graphical program was generated by the GPG program.

Cited [0120] discusses the fact that the GPG program may be associated with a program or application, such as a graphical program development environment, that may directly aid the user in creating a new graphical program.

Cited [0020] discloses that the GPG program may be constructed using any of various programming technologies or techniques, and may be a text-based program and/or a graphical program.

Cited Figure 6 and [0136] – [0140] discuss the GPG program generating a graphical program in response to initial program information, then modifying the graphical program based on subsequent program information.

Appellant respectfully notes that:

1) the GPG program generates graphical program code for a *new* graphical program, not for itself. For example, the cited New VI Object Reference node in the GPG program is configured to create a new VI object (e.g., based on a VI object style input value), such as, for example, the cited waveform chart control and waveform chart user interface node, then when the GPG program is run, this New VI Object Reference node executes and creates a new VI object of the specified style in a *new graphical program*; and

2) the program code that creates the specified new VI object at runtime is inherently already part of the New VI Object Reference node, and, per 1), any graphical program code generated by the New VI Object Reference node belongs to the new graphical program.

Thus, in Kudukoli, there is no *associating* determined code with the New VI Object Reference node. Nowhere does Kudukoli mention or even hint at determining graphical program code for a node based on user-selection of displayed functions for the node, and *associating the determined graphical program with the node, where when the node executes, the determined graphical program code executes to perform the function*.

In fact, Kudukoli nowhere describes associating determined graphical program code with a node that is already displayed in a graphical program at all. The only associating Kudukoli discusses is between the GPG program and the received program information ([0037], [0147]), between the GPG program and a program or application that aids the user in creating the GPG program or a new graphical program, e.g., a development environment or application ([0022] and elsewhere), and between a new graphical program and a new programming environment ([0023] and elsewhere).

Thus, Appellant submits that even if Kudukoli's New VI Object Reference node code that generates a new VI object for new graphical program were considered to be Appellant's claimed "determined graphical program code", this code (of the New VI Object Reference node) inherently belongs to the New VI Object Reference node, and

thus no *associating* of this code with the node is performed. Nowhere does Kudukoli describe or even hint at associating determined graphical program code with the New VI Object Reference node based on selected functionality for the node. Thus, the Advisory Action's assertion that Kudokoli teaches associating code for generating the waveform chart (or random number generator or stop button) with the node is incorrect, since this code already belongs to the node.

The Examiner has improperly read this claimed feature into Kudokoli without presenting any evidence to support the feature. Cited [0278]-[0279] and [0282] (and Kudukoli in general) in no way disclose this feature, nor does Kudokoli indicate anywhere that this code is graphical program code. Appellant respectfully requests that the Examiner particularly describe and explain where these features may be found in Kudokoli. Note that the Examiner's assertion that Kudukoli teaches, "if 'waveform chart' selected [*sic*], associating determined graphical program code with said 'waveform' control/node", citing [0278], and thus teaches this feature is clearly incorrect–the waveform chart/control (in the front panel) and the waveform chart user interface node (in the block diagram of the new graphical program), already have their own code (which is never referred to as being graphical program code, and which is never described as being determined in response to selection of a function and associated with the waveform chart control/node), nor is the code implementing these control/node elements executed when the New VI Object Reference node executes; rather, this code executes when the *new* graphical program executes. Additionally, Appellant notes that the "waveform chart user interface control" which is generated by the New VI Object Reference node does not in fact have the appearance of the New VI Object Reference node, contrary to the Examiner's assertion.

Similarly, as Appellant notes above, any new VI object created by the New VI Object Reference node *belongs to the new graphical program*, and its code is neither associated with the New VI Object Reference node, nor executed when the New VI Object Reference node executes; rather, the new VI object is created by and when the New VI Object Reference node is executed. Appellant submits that in the Advisory Action, the Examiner appears to be blurring the distinction between the inherent code of the New VI Object Reference node that creates new nodes/objects for the new graphical

program, and the created nodes/objects themselves, which is improper and incorrect. Moreover, the Examiner appears to be ignoring the fact that in Kudukoli the New VI Object Reference node is in a first graphical program, while the generated new VI object is in a different (automatically generated) graphical program. This is in direct contrast with Appellant's invention as recited in claim 69, where there is only one graphical program that includes the original node, both before and after the claimed associating.

Thus, Appellant respectfully submits that Kudukoli fails to disclose all the features and limitations of claim 69, and so, for at least the reasons provided above, Appellant submits that claim 69 is patentably distinct and non-obvious over Kudukoli, and thus allowable.

Claim 70, 71, 78, 79

In addition to the novel limitations of claim 69, Kudukoli fails to teach or suggest **wherein the node has a first node icon which is displayed in the graphical program, and wherein the first node icon has a first appearance, wherein the program instructions are further executable to perform: changing the first node icon to a second appearance based on the second user input, wherein said changing the first node icon to a second appearance includes displaying an image corresponding to the selected function**, as recited in claim 70.

In asserting that Kudukoli teaches these features, the Examiner cites [0130], [0269], and [0030].

Regarding the limitation "changing the first node icon to a second appearance based on the second user input", Appellant notes that cited [0030] and [0130] each discusses the fact that the GPG program may be used to convert a first graphical program to a new graphical program, e.g., under a different programming environment, where the new graphical program has a visual appearance similar to the original program, or the GPG program may generate the new graphical program such that it may have a different appearance than the original program. Note that these citations are directed to the appearance of a generated graphical program, *not* the appearance of individual node icons, and so are not germane to this feature. Thus, for at least these reasons, these citations fail to teach or suggest this claimed limitation.

Regarding the limitation "wherein said changing the first node icon to a second appearance includes displaying an image corresponding to the selected function", cited [0269] describes a VI refnum, specifically, a VI Server front panel refnum control being placed in a front panel (a GUI for a graphical program), and configured to be some specified type of refnum, after which the refnum control us the appearance of the specified type of refnum. First, Appellant notes that a refnum control is not a graphical program node for use in a graphical program, but rather is a type of control for use in a front panel (GUI) that provides a reference to an appropriate entity, e.g., to an application, VI, etc. Thus, this citation fails to teach or suggest changing a first (graphical program) node icon to a second appearance, including displaying an image corresponding to the selected function.

Thus, Kudukoli fails to teach these features of claim 70, and so claim 70 is patentably distinct and nonobvious, and thus allowable.

Claim 72, 80

In addition to the novel limitations of claim 69, Kudukoli fails to teach or suggest **wherein, prior to said associating the determined graphical program code with the node, the node does not have any associated graphical program code**, as recited in claim 72.

Cited [0225] is directed to an input of an Upcast Reference node, specifically, a "vi object class" input that specifies a class to upcast the object reference to. This has nothing whatsoever to do with a node not having any associated graphical program code prior to associating determined graphical program code with the node.

Similarly, cited [0230] is directed to an input of a Downcast Reference node, specifically, a "vi object class" input that specifies a class to downcast the object reference to, which also has nothing whatsoever to do with a node not having any associated graphical program code prior to associating determined graphical program code with the node.

Thus, Kudukoli fails to teach these features of claim 71, and so claim 71 is patentably distinct and nonobvious, and thus allowable.

Claim 73, 81

In addition to the novel limitations of claim 69, Kudukoli fails to teach or suggest **wherein, prior to said associating the determined graphical program code with the node, the node has associated default graphical program code in accordance with a default function for the node, and wherein the default graphical program code implements a first functionality; and wherein said associating the determined graphical program code with the node comprises replacing the default graphical program code with the determined graphical program code**, as recited in claim 73.

Regarding the limitation "wherein, prior to said associating the determined graphical program code with the node, the node has associated default graphical program code in accordance with a default function for the node, and wherein the default graphical program code implements a first functionality", cited [0225] is directed to an input of an Upcast Reference node, specifically, a "vi object class" input that specifies a class to upcast the object reference to, as explained above. Nowhere does this citation, nor Kudukoli in general, indicate that the Upcast Reference node has associated default graphical program code, nor any kind of graphical program code at all. Appellant submits that the Examiner has improperly speculated as to the particular implementation of these nodes, absent any supporting evidence.

Regarding the limitation "wherein, prior to said associating the determined graphical program code with the node, the node has associated default graphical program code in accordance with a default function for the node, and wherein the default graphical program code implements a first functionality", cited [0217] describes an input to the New VI Object Reference node, specifically, a "style" input that specifies the style or subclass of an object to be created by the new VI Object Reference node upon execution. Note that neither this citation, nor Kudukoli in general, indicates that the New VI Object Reference node includes graphical program code.

Cited [0278] is directed to using the New VI Object Reference node to create the waveform charge user interface control as part of creating a new graphical program. Note that a reference to the new graphical program is connected as the "owner reference" in of the New VI Object Reference node, and that after the GPG program has executed the New VI Object Reference node to create the new graphical program, including the

waveform chart user interface control, the New VI Object Reference node outputs a reference to the waveform control. Note that the New VI Object Reference node's underlying code does not change, and does not require an "association" operation to belong to the New VI Object Reference node. Nor does the code of the generated waveform chart user interface control require such an operation. More specifically, Appellant notes that the New VI Object Reference node nor the generated waveform chart user interface control nor the waveform chart user interface node has associated default graphical program code in accordance with a default function for the node, nor is the code for these elements replaced with determined graphical program code in response to user selection of functionality for these elements. Thus, the citation does not disclose this claimed feature.

Thus, Kudukoli fails to teach these features of claim 73, and so claim 73 is patentably distinct and nonobvious, and thus allowable.

Claim 74, 82

In addition to the novel limitations of claim 69, Kudukoli fails to teach or suggest **wherein said receiving first user input comprises receiving the first user input to the node; and wherein said receiving second user input comprises receiving the second user input to the node**, as recited in claim 74.

Cited Figure 21 and related text are directed to users specification of the vi object class input via a hierarchical menu. However, Appellant notes that Kudukoli nowhere indicates that this menu is invoked by user input to the node itself.

Thus, Kudukoli fails to teach these features of claim 74, and so claim 74 is patentably distinct and nonobvious, and thus allowable.

Claim 75, 83

In addition to the novel limitations of claim 69, Kudukoli fails to teach or suggest **wherein said displaying the plurality of functions for the node in response to the first user input comprises: displaying a plurality of function classes for the node; and in response to user input selecting a function class, displaying the plurality of**

functions, wherein the plurality of functions are in the selected function class, as recited in claim 75.

As noted above, cited [0217] describes an input to the New VI Object Reference node, specifically, a "style" input that specifies the style or subclass of an object to be created by the new VI Object Reference node upon execution. Appellant notes that a style or subclass of an object to be created is not equivalent to a function class, i.e., a class of functions, and respectfully submits that the Examiner as improperly equated the object oriented concept of an object class with the more general notion of a class of functions, i.e., a category or group of functions.

As also noted above, cited [0278] is directed to using the New VI Object Reference node to create the waveform charge user interface control as part of creating a new graphical program. Nowhere does this citation disclose displaying a plurality of *function classes* for a node in response to user input, nor displaying a plurality of functions of that class in response to selection of a function class. Appellant further notes that Figure 25A, to which this text refers, also fails to show these features.

Appellant further notes that Figures 21 and 22 illustrate hierarchical menus whereby the user selects front panel controls to be created by the New VI Object Reference node, and that the displayed options correspond to object types or classes that the New VI Object Reference node can create, *not* classes of functions and functions in those classes.

Thus, Kudukoli fails to teach these features of claim 75, and so claim 75 is patentably distinct and nonobvious, and thus allowable.

Claim 76, 84

In addition to the novel limitations of claim 69, Kudukoli fails to teach or suggest **wherein the node is a data acquisition (DAQ) node; wherein the plurality of functions for the node comprise a plurality of DAQ functions; wherein, prior to said associating, the DAQ node comprises one of:**

a generic read node;

a generic write node;

a generic channel creation node;

a generic timing node; or

a generic triggering node; and

wherein, after said associating, the DAQ node comprises one of:

a specific read node in accordance with the selected function;

a specific write node in accordance with the selected function;

a specific channel creation node in accordance with the selected function;

a specific timing node in accordance with the selected function; or

a specific triggering node in accordance with the selected function, as recited in claim 76.

Cited [0015] describes graphical programming in general, but makes no mention of a graphical program DAQ node as claimed. Similarly, [0085] is directed to an exemplary instrumentation control system (shown in Figure 2A), but similarly fails to disclose such a DAQ node.

Cited [0080] describes exemplary instruments in the exemplary instrumentation control system of Figure 2A, and does not disclose or describe, or even hint at, a data acquisition (DAQ) node that includes a plurality of DAQ functions as claimed.

Cited [0092] discusses the GPG program, noting that this program and/or the graphical program generated by the GPG program may be designed for various functionalities, including for example, data acquisition/generation, analysis, and/or display, and for controlling or modeling instrumentation or industrial automation hardware. However, this citation makes no reference to a DAQ node with a plurality of DAQ functions.

As noted above, cited [0085] is directed to an exemplary instrumentation control system (shown in Figure 2A), but makes no mention of a DAQ node that, prior to associating determined graphical program code with the node, is a generic read node, a generic write node, a generic channel creation node, a generic timing node, or a generic triggering node, as claimed.

Similarly, cited [0092] and [0015], described above, in no way disclose a DAQ node that, after associating determined graphical program code with the node, is a specific read node in accordance with the selected function, a specific write node in

accordance with the selected function, a specific channel creation node in accordance with the selected function, a specific timing node in accordance with the selected function, or a specific triggering node in accordance with the selected function, as claimed.

Nowhere does Kukukoli disclose this claimed conversion of a generic function node to a specific function node in accordance with a user-selected function for the node.

Thus, Kudukoli fails to teach these features of claim 76, and so claim 76 is patentably distinct and nonobvious, and thus allowable.

Claims 85, 89

Independent claim 85 is separately patentable because the cited reference does not teach or suggest the limitations recited in this claim. For example, Appellant respectfully submits that Kudukoli fails to disclose **determining a second node based on the selected function, wherein the second node comprises a graphical representation of an implementation of the selected function, and wherein the second node comprises graphical program code executable to provide functionality in accordance with the selected function**.

Cited [0022] describes associating the GPG program with a program or application that aids the user in creating the GPG program or a new graphical program, e.g., a development environment, application, or wizard, but makes no mention of determining a second node based on a function selected by a user for a first node in a graphical program, where the second node is executable to perform the selected function. Note, for example, that Kudukoli's New VI Object Reference Node generates a new VI object based on inputs to the node, i.e., the input, e.g., user selection of object style, specifies what object the New VI Object Reference Node will create when executed. The new VI object is *not* executable to perform the selected functionality, that is, the creation functionality that generates the new VI object. Cited [0029] discusses dependence of the functionality of Kudukoli's generated graphical program on both received information and the GPG program. Again, this citation makes no mention of determining a second node based on a function selected by a user for a first node in a graphical program, where the second node is executable to perform the selected function. Finally, cited [0030]

describes the "typical case" where "the implementation of the graphical program code is determined mainly or entirely by the GPG program, although the received information may influence the manner in which the GPG program generates the code, or the GPG program may receive separate information influencing the code generation." More specifically, this citation discloses the GPG program translating an existing graphical program to a new graphical program.

Applicant submits that in the above citations, and Kudukoli in general, no description is provided of determining a second node based on user-selection of a function for a first node in a graphical program, where the second node is executable to perform the selected function. The Advisory Action's assertion that "the waveform control/node (the second node) has replaced the New VI Object Reference Node (the first node) by having the same icon/appearance with the New VI Object Reference Node but with the determined graphical program code associated with 'waveform chart' [*sic*]" is incorrect. As noted above, the code that creates the waveform chart is *inherent* in the New VI Object Reference node and so no *associating* is performed, and furthermore, is nowhere described as graphical program code. Additionally, the waveform control is not a graphical program node, but rather a GUI element, and the created waveform chart user interface node is in the new graphical program, and thus does not replace the New VI Object Reference node. Nor does the waveform chart user interface node, which is generated by the New VI Object Reference node, replace the New VI Object Reference node in the GPG program–rather, the waveform chart user interface node is placed in the new graphical program.

Similarly, the cited "random number generator control/node" ([0282]), which is actually referred to as "the random number generator function" node, the cited wait function node, and the cited NOT Boolean node, are all created (by the New VI Object Reference node) and inserted into the *new* graphical program, and thus do not replace the New VI Object Reference node. Again, Applicant submits that in the Advisory Action, the Examiner appears to be blurring the distinction between the inherent code of the New VI Object Reference node that creates new nodes/objects for the new graphical program, and the created nodes/objects themselves, which is improper and incorrect–the created nodes do not replace the New VI Object Reference node.

For example, in the Advisory Action, the Examiner asserts that since claim 90 (the Examiner actually refers to claim 96, which doesn't exist) recites "wherein the node comprises a first node icon, and wherein said displaying the node comprises displaying the first node icon, and wherein the second node comprises: the first node icon and the graphical program code; or a second node icon and the graphical program code", that this is somehow taught by Kudukoli in that "the second node may comprise the icon/appearance of the first node (New VI Object Reference Node) and the determined graphical program code associated with first and second user inputs (associated with 'waveform chart', 'stop button', or 'random number generator'). [*sic*]". Appellant notes that the 'waveform chart', 'stop button', or 'random number generator' do not replace the New VI Object Reference Node (which, Appellant notes, is never replaced) in the GPG program, nor does the code for these GUI elements (which is nowhere disclosed as graphical program code) execute when the New VI Object Reference Node executes, nor are these GUI elements in the same graphical program as the New VI Object Reference Node.

Thus, for at least these reasons, the cited art does not teach or suggest these claimed features.

Additionally, nowhere does Kudukoli disclose **replacing the node in the graphical program with the second node, wherein, when the second node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function**, as recited in claim 85

Cited Figure 22 illustrates how a user may choose a value for the style input by selecting from a hierarchical menu, e.g., invoked with respect to a New VI Object Reference node. Note that once configured, the New VI Object Reference node executes in a GPG program to create a new VI object for or in *another graphical program*, as discussed above. The created object does not replace the New VI Object Reference node in the GPG program. In other words, no node is replaced by the created VI object generated by the New VI Object Reference node.

Paragraph [0225] discusses the VI object class input to an Upcast Reference node, which specifies a class to which the Upcast Reference node will cast a VI object

reference. Paragraph [0230] discusses the VI object class input to a Downcast Reference node, which specifies a class to which the Downcast Reference node will cast a VI object reference. Neither of these citations mentions or even hints at replacing a node in a graphical program with a second node that executes determined graphical program code as claimed.

Applicant has reviewed Kudukoli closely, and respectfully submits that no mention is made in the entire reference of performing Applicant's claimed node replacement in a graphical program based on user-selected functionality for the node.

Thus, or at least the reasons provided above, Applicant submits that Kudukoli fails to teach or suggest these features and limitations of claim 85, and so claim 85 and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claim 86, 90

In addition to the novel limitations of claim 85, Kudukoli fails to teach or suggest **wherein the node comprises a first node icon, and wherein said displaying the node comprises displaying the first node icon, and wherein the second node comprises: the first node icon and the graphical program code; or a second node icon and the graphical program code**, as recited in claim 86.

In asserting that Kudukoli teaches these features, the Examiner cites [0215]-[0216]. [0215] is directed to a "vi object class" input to the New VI Object Reference node that specifies the type of object to create, and [0216] discloses an "owner reference" input to the New VI Object Reference node that specifies a reference to the VI or VI object that will "own" or "contain" the new object. However, neither citation teaches or suggests first and second function nodes as claimed, i.e., where displaying the first function node includes displaying a first node icon, and where the first node is replaced by a second node that includes the first node icon and the automatically determined graphical program code (corresponding to a selected function), or the determined graphical program code and a second node icon. In fact, as discussed above at length, Kudukoli nowhere discloses such determination of graphical program code for a node at all.

Thus, or at least the reasons provided above, Applicant submits that Kudukoli fails to teach or suggest these features and limitations of claim 86, and so claim 86 is patentably distinct and non-obvious over the cited art, and thus allowable.

Claim 87, 91

In addition to the novel limitations of claim 85, Kudukoli fails to teach or suggest **wherein the node and/or the second node is one or more of: polymorphic; function switchable; or function class switchable**, as recited in claim 87.

In asserting that Kudukoli teaches these features, the Examiner cites [0217] and [0278]. As discussed above, [0217] is directed to an input to the New VI Object Reference node, specifically, a "style" input that specifies the style or subclass of an object to be created by the new VI Object Reference node upon execution. Appellant notes that a style or subclass of an object to be created is not equivalent to a function class, i.e., a class of functions, nor do the object creation types/subclasses of the new VI Object Reference node comprise functions for the node–rather, these options allow the user to specify the type of object to be created by the node upon execution. Nor does Kudukoli ever mention function nodes that are polymorphic. In fact, the only mention Kukukoli makes of "polymorphic" is with regard to the "VI name or path" input of an Open VI Reference node in [0189], which is not at all the same thing.

Thus, or at least the reasons provided above, Applicant submits that Kudukoli fails to teach or suggest these features and limitations of claim 87, and so claim 87 is patentably distinct and non-obvious over the cited art, and thus allowable.

Claim 88, 92

In addition to the novel limitations of claim 85, Kudukoli fails to teach or suggest **wherein the node is a data acquisition (DAQ) node; wherein the DAQ node comprises one of:**

a generic read node;

a generic write node;

a generic channel creation node;

a generic timing node; or

a generic triggering node; and

wherein the second node comprises a corresponding one of:

a specific read node in accordance with the selected function;

a specific write node in accordance with the selected function;

a specific channel creation node in accordance with the selected function;

a specific timing node in accordance with the selected function; or

a specific triggering node in accordance with the selected functio, as recited in claim 88.

Cited [0015] describes graphical programming in general, but makes no mention of a graphical program DAQ node as claimed. Similarly, [0085] is directed to an exemplary instrumentation control system (shown in Figure 2A), but similarly fails to disclose such a DAQ node.

Cited [0080] describes exemplary instruments in the exemplary instrumentation control system of Figure 2A, and does not disclose or describe, or even hint at, a data acquisition (DAQ) node that includes a plurality of DAQ functions as claimed.

Cited [0092] discusses the GPG program, noting that this program and/or the graphical program generated by the GPG program may be designed for various functionalities, including for example, data acquisition/generation, analysis, and/or display, and for controlling or modeling instrumentation or industrial automation hardware. However, this citation makes no reference to a DAQ node with a plurality of DAQ functions.

As noted above, cited [0085] is directed to an exemplary instrumentation control system (shown in Figure 2A), but makes no mention of a DAQ node that, prior to associating determined graphical program code with the node, is a generic read node, a generic write node, a generic channel creation node, a generic timing node, or a generic triggering node, as claimed.

Similarly, cited [0092] and [0015], described above, in no way disclose a DAQ node that, after associating determined graphical program code with the node, is a specific read node in accordance with the selected function, a specific write node in accordance with the selected function, a specific channel creation node in accordance

with the selected function, a specific timing node in accordance with the selected function, or a specific triggering node in accordance with the selected function, as claimed.

Nowhere does Kukukoli disclose this claimed conversion of a generic function node to a specific function node in accordance with a user-selected function for the node.

Thus, Kudukoli fails to teach these features of claim 88, and so claim 88 is patentably distinct and nonobvious, and thus allowable.

For the foregoing reasons, it is respectfully submitted that the Examiner's rejection of claims 69-92 was erroneous, and reversal of the decision is respectfully requested.

The fee of $510.00 for filing this Appeal Brief is being paid concurrently via EFS-Web. If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-80201/JCH.

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons Hood Kivlin Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: 2008-08-05 JCH/MSW

VIII. CLAIMS APPENDIX

The following lists claims 69-92, incorporating entered amendments, as on appeal.

69. A computer-accessible memory medium that stores program instructions executable by a processor to perform:

displaying a node in a graphical program;

receiving first user input invoking display of a plurality of functions for the node;

displaying the plurality of functions for the node in response to the first user input;

receiving second user input selecting a function from the plurality of functions;

determining graphical program code based on the second user input, wherein the determined graphical program code comprises a graphical representation of an implementation of the selected function, and wherein the determined graphical program code is executable to provide functionality in accordance with the selected function;

associating the determined graphical program code with the node, wherein, when the node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function.

70. The memory medium of claim 69, wherein the node has a first node icon which is displayed in the graphical program, and wherein the first node icon has a first appearance, wherein the program instructions are further executable to perform:

changing the first node icon to a second appearance based on the second user input, wherein said changing the first node icon to a second appearance includes displaying an image corresponding to the selected function.

71. The memory medium of claim 70,

wherein said changing the first node icon to a second appearance comprises replacing the first node icon with a second node icon.

72. The memory medium of claim 69, wherein, prior to said associating the determined graphical program code with the node, the node does not have any associated graphical program code.

73. The memory medium of claim 69,

wherein, prior to said associating the determined graphical program code with the node, the node has associated default graphical program code in accordance with a default function for the node, and wherein the default graphical program code implements a first functionality; and

wherein said associating the determined graphical program code with the node comprises replacing the default graphical program code with the determined graphical program code.

74. The memory medium of claim 69,

wherein said receiving first user input comprises receiving the first user input to the node; and

wherein said receiving second user input comprises receiving the second user input to the node.

75. The memory medium of claim 69,

wherein said displaying the plurality of functions for the node in response to the first user input comprises:

displaying a plurality of function classes for the node; and

in response to user input selecting a function class, displaying the plurality of functions, wherein the plurality of functions are in the selected function class.

76. The memory medium of claim 69,

wherein the node is a data acquisition (DAQ) node;

wherein the plurality of functions for the node comprise a plurality of DAQ functions;

wherein, prior to said associating, the DAQ node comprises one of:

a generic read node;

a generic write node;

a generic channel creation node;

a generic timing node; or

a generic triggering node; and

wherein, after said associating, the DAQ node comprises one of:

a specific read node in accordance with the selected function;

a specific write node in accordance with the selected function;

a specific channel creation node in accordance with the selected function;

a specific timing node in accordance with the selected function; or

a specific triggering node in accordance with the selected function.

77. A computer-implemented method for configuring a graphical program node, comprising:

displaying a node in a graphical program;

receiving first user input invoking display of a plurality of functions for the node;

displaying the plurality of functions for the node in response to the first user input;

receiving second user input selecting a function from the plurality of functions;

determining graphical program code based on the second user input, wherein the determined graphical program code comprises a graphical representation of an implementation of the selected function, and wherein the determined graphical program code is executable to provide functionality in accordance with the selected function;

associating the determined graphical program code with the node, wherein, when the node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function.

78. The method of claim 77, wherein the node has a first node icon which is displayed in the graphical program, the method further comprising:

changing the first node icon to a second appearance based on the second user input, wherein said changing the first node icon to a second appearance includes displaying an image corresponding to the selected function.

79. The method of claim 78,

wherein said changing the first node icon to a second appearance comprises replacing the first node icon with a second node icon.

80. The method of claim 77, wherein, prior to said associating the determined graphical program code with the node, the node does not have any associated graphical program code.

81. The method of claim 77,

wherein, prior to said associating the determined graphical program code with the node, the node has associated default graphical program code in accordance with a default function for the node, and wherein the default graphical program code implements a first functionality; and

wherein said associating the determined graphical program code with the node comprises replacing the default graphical program code with the determined graphical program code.

82. The method of claim 77,

wherein said receiving first user input comprises receiving the first user input to the node; and

wherein said receiving second user input comprises receiving the second user input to the node.

83. The method of claim 77,

wherein said displaying the plurality of functions for the node in response to the first user input comprises:

displaying a plurality of function classes for the node; and

in response to user input selecting a function class, displaying the plurality of functions, wherein the plurality of functions are in the selected function class.

84. The method of claim 77,

wherein the node is a data acquisition (DAQ) node;

wherein the plurality of functions for the node comprise a plurality of DAQ functions;

wherein, prior to said associating, the DAQ node comprises one of:

a generic read node;

a generic write node;

a generic channel creation node;

a generic timing node; or

a generic triggering node; and

wherein, after said associating, the DAQ node comprises one of:

a specific read node in accordance with the selected function;

a specific write node in accordance with the selected function;

a specific channel creation node in accordance with the selected function;

a specific timing node in accordance with the selected function; or

a specific triggering node in accordance with the selected function.

85. A computer-accessible memory medium that stores program instructions executable by a processor to perform:

displaying a node in a graphical program;

receiving first user input invoking display of a plurality of functions for the node;

displaying the plurality of functions for the node in response to the first user input;

receiving second user input selecting a function from the plurality of functions;

determining a second node based on the selected function, wherein the second node comprises a graphical representation of an implementation of the selected function,

and wherein the second node comprises graphical program code executable to provide functionality in accordance with the selected function;

replacing the node in the graphical program with the second node, wherein, when the second node in the graphical program executes, the graphical program code of the second node executes to provide the functionality in accordance with the selected function.

86. The memory medium of claim 85, wherein the node comprises a first node icon, and wherein said displaying the node comprises displaying the first node icon, and wherein the second node comprises:

the first node icon and the graphical program code; or

a second node icon and the graphical program code.

87. The memory medium of claim 85, wherein the node and/or the second node is one or more of:

polymorphic;

function switchable; or

function class switchable.

88. The memory medium of claim 85,

wherein the node is a data acquisition (DAQ) node;

wherein the DAQ node comprises one of:

a generic read node;

a generic write node;

a generic channel creation node;

a generic timing node; or

a generic triggering node; and

wherein the second node comprises a corresponding one of:

a specific read node in accordance with the selected function;

a specific write node in accordance with the selected function;

a specific channel creation node in accordance with the selected function;

a specific timing node in accordance with the selected function; or

a specific triggering node in accordance with the selected function.

89. A computer-implemented method for configuring a graphical program node, comprising:

displaying a node in a graphical program;

receiving first user input invoking display of a plurality of functions for the node;

displaying the plurality of functions for the node in response to the first user input;

receiving second user input selecting a function from the plurality of functions;

determining a second node based on the selected function, wherein the second node comprises a graphical representation of an implementation of the selected function, and wherein the second node comprises graphical program code executable to provide functionality in accordance with the selected function;

replacing the node in the graphical program with the second node, wherein, when the second node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function.

90. The method of claim 89, wherein the node comprises a first node icon, and wherein said displaying the node comprises displaying the first node icon, and wherein the second node comprises:

the first node icon and the graphical program code; or

a second node icon and the graphical program code.

91. The memory medium of claim 89, wherein the node and/or the second node is one or more of:

polymorphic;

function switchable; or

function class switchable.

92. The memory medium of claim 89,

wherein the node is a data acquisition (DAQ) node;

wherein the DAQ node comprises one of:

a generic read node;

a generic write node;

a generic channel creation node;

a generic timing node; or

a generic triggering node; and

wherein the second node comprises a corresponding one of:

a specific read node in accordance with the selected function;

a specific write node in accordance with the selected function;

a specific channel creation node in accordance with the selected function;

a specific timing node in accordance with the selected function; or

a specific triggering node in accordance with the selected function.

IX. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

X. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.